Lightweight Image-based Key Point Tracking For Real-Time Bridge Monitoring With Smartphones

University of Twente
PO Box 217, 7500 AE Enschede
The Netherlands

ABSTRACT

Key point detection has become more relevant in many industries. From maintenance to development of materials. The usage of this technology is not equally distributed. Some sectors have been more integrated with this technique than others. The bridge maintenance sector still uses IMU(Inertial movement unit) sensors for their inspection of critical points on bridges. This is quite outdated since for large bridges many IMU sensors are needed. We made an application which can estimate the movement of critical points using a video taken from a smartphone. This will aid in more efficient performance and lower the cost of bridge maintenance. Since less to no IMU sensors will be needed to perform the check up. As IMU sensors can take a couple of hours to be placed and removed from the bridge. Their data needs to be progressed and calibrated which could take a couple. In the future, this application could replace the IMU sensors completely and result in efficient and less time consuming bridge maintenance.

Keywords

Bridge detection, key point detection tool, IMU sensors and keypoint movement.

1. INTRODUCTION

Bridges have been around for centuries. The use of the bridge has not declined in recent years. The digital revolution has not completely integrated into every sector of the bridge, from design to production to maintenance. The maintenance part of the bridge especially on the estimation of movement on the critical points can be improved.

Currently IMU sensors are used for the estimation for the critical points movement. The IMU sensor is placed on a specific point of the bridge. After turning the sensor on. It measures the vibration of the object using the built in accelerometer and gyroscope. The data is saved inside the IMU sensor. The data can be used to check the vibration over the entire object and the key points. This is quite an outdated practice. Since large bridges need a large number of IMU sensors and time to place them. For example a large bridge needs multiple IMU sensors placed on the bridge. In order to measure the movement. The sensors need to be placed. To place and retrieve the sensors may take a couple of hours or more or less depending on the object. Afterwards the IMU sensors need to be retrieved to read out the data. The data from the IMU sensors need to be processed and calibrated with the computer in order to read the data. Which could take an extra couple of hours. This new era grants more possibilities to decrease the time spent on the task.

If a mobile phone were able to process a video of a bridge to determine the key points movement. Time can be spared on placing and retrieving the IMU sensor as the application only requires the video. Which improves the efficiency to complete the task immensely. There are some algorithms and functions made to track the individual components of the application For example Optical Flow is used to track key points in videos. Key points selection can be done with the use of object detection. Key point detection can be done using the Shi Tomasi Corner Detection.

For the creation of the application each segment was turned into a goal:

- **Goal 1:** To create a key point detection tool, which can process the first frame of the video and place the key points.
- **Goal 2:** To filter the key points from goal 1. So, only the bridge key points remain. Goal 1 output will be used.
- **Goal 3**: To create a python function which will track the movement of the key points and return the movement.
- **Goal 4:** To create a program to separate the movement of the camera and the movement of the bridge.

In order to complete the goals the following research questions were made for our project:

- **Research Question 1:**
  What kind of keypoint detection tool is needed to detect the key points in the image?
- **Research Question 2:**
  What kind of operation is needed to filter the key points of the image so that the key points of the bridge remain?
- **Research Question 3:**
  How to estimate the bridge keypoints movement per frame?
- **Research Question 4:**
  How to check if the application estimations are correct?
- **Research Question 5:**
  What kind of implementation with the use of a fixed point chosen by the user in the first frame is needed to stabilize the video?

Our research contributes in the field of key points detection and movement tracking for maintenance of bridges. The result is an application which can track the movement of a bridge key points and determine the max movement.

The structure of this thesis has been divided into sections. Section 2 will show related work in the department of keypoint selection, object detection, keypoint tracking and camera calibration. Section 3 will explain the methodology for each research question. Section 4 explains the result in detail. Section 5 will give a conclusion of the research. With a short discussion about the research in section 6.

2. RELATED WORK

The following section will show related work in image alignment or key point detection.

In 1994 a paper for an improved Harris Corner Detection by J. Shi and C. Tomasi was introduced. The paper showed an improved version of the Harris Corner detection which improved the corner detected by changing the formula inside the Harris Corner Detection. The resulting corners were better to track from frame to frame.[17]

In the paper Consistent image alignment for video mosaicing published online in 2011. A new method of image alignment for mosaicing is proposed. With the use of homography. The result of this research was a globally consistent and accurate image alignment.[18]

Pattern Recognition and Image Analysis was published in 2013. The book contains information about the analysis of image recognition and pattern analysis.

In 2017, research was performed to make key point detection in videos more efficient to estimate poses.[14] The result of their research was an efficient method to make human keypoint in videos.

In 2017 a book was published to process an image to a geometric view. The result view is beneficial for the computer. Since it reduces information to a core minimum.[16]

In 2018 research was performed to propose a new key frame selection method.[13] In order to view the key points of videos within a certain time frame.

Research was published in 2018 to detect key components of existing bridges in point cloud datasets.[15] By automating the modelling of existing bridges will improve efficiency and cost.

In 2019 a research paper was published for advanced selective key point detection techniques.[11] The researchers tried to create an advanced selective key point detection from a combination of hypotheses of the subject.

In 2020 Yolov4 paper was released for object detection. The researchers improved the previous version of Yolov3. The improvement was enough to be called the successor of Yolov3[12] .

3. METHODOLOGIES

This section will be used to describe the steps which were taken to create the application.
As explained in the introduction the app will first calibrate the camera removing the movement of the hand. Afterwards it will detect the key points of the video. The first frame of the video will be used to detect the key points. Afterwards the key points need to be filtered. With the use of object detection. The bridge is detected and the bounding box edges are saved in a .txt file. These will be used to filter the keypoints. The remaining key points after filtering are on the bridge. Afterwards the movement of the keypoints is tracked using the Optical Flow algorithm. The movement per keypoints is afterwards displayed in the app. The app will be tested by using a wooden bridge video made at the University of Twente. Where Imu sensors are placed.
Due to time constraints it was not possible to have an advanced application with self made code. Certain parts of the project were made using code available on github and online tutorials.
The application was made using primarily Python.

3.1 Data Source

The following photos of bridges were collected from the website.[10] The photos are free usable images. Around 200 photos were collected from the site. Image augmentation was used to create an additional 300 photos. The augmentation photos differ from the original by applying the following operation: rotating, cropping or/and grayscaling. The images were collected and saved as .jpg files. For annotating the images a labeling website was used to create the annotation for yolov4 format.[2] For training the model AlexeyAB repository was used from github.[6] For object detection and conversion of the yolov4 weights to tensorflow lite the following github repository was used.[7]

3.2 Dataset

The total photos set contains 500 images. Where 300 images were created using data augmentation on the original 200 bridges. Every photo contains its a .txt file with the bounding boxes for the object detention. For either training or testing. The .txt files were created from the labeling website.[2] The dataset was split into a training and testing data set. The data for training contained augmented images while the other dataset contained non-augmented images.

3.3 Research Question 1

After some comparing algorithms online. The method goodFeaturesToTrack() was chosen. The method is an opencv python library function to detect the key points in the first frame of the video.[5]

The function originates from the opencv library. The function implements the Shi-Tomasi corner detection. The Shi-Tomasi is an improvement of the Harris Corner detection. The Harris Corner Detector filters the points if they are a corner, an edge or flat region.[4]

The following parts explain the Harris Corner Detection. The function in Figure 1 is used to calculate the variation intensity of in the window w(u,v).[4] The window function is either a rectangular window or a Gaussian window which gives weights to pixels underneath. [4] Afterwards to maximize the function E(u,v) the second term needed to be maximized. With the use of Taylor expansion the following equation in Figure 2 can be derived. The function M is defined in Figure 3. Where Ix and Iy are image derivatives of x and y directions. Finally the main equation in Figure 4 is used to determine the score of the point. Where:

- $\det(M)=\lambda 1\lambda 2$,
- $\text{trace}(M)=\lambda 1+\lambda 2$

The regions of approval can be observed in Figure 5.

Shi-Tomasi improved the function by rewriting the R from Harris Corner Detector displayed Figure 6.[5] To the function R observed in Figure 7. Which results into a different region of approval as shown in Figure 8:

Compared to the other corner if $\lambda 1$ and $\lambda 2$ are above the λ minimum the value is a corner. The corners which are below a certain quality λ are rejected. This will result in the best corners also known as key points to track for Optical Flow.[5]

3.4 Research Question 2

For the operation for filtering the key points. Only the key points of the bridge should remain. Yolov4 was used to filter the key points. Yolov4 is an object detection algorithm which draws bounding boxes on the image on the object. Due to the fast processing speed Yolov4 was chosen for training the custom object detection for the bridge. The results of the custom object detection training were the weights. These were converted to tensorflow lite weights. Since object detection on smartphones is better run on tensorflow lite since it takes less computing power. The function works as follows. An image is put in with a bridge or not. If the application detects the bridge. A bounding box is drawn as shown in Figure 9. If no bridge is detected no bounding box is drawn.

3.5 Research Question 3

After determining the key points. The key points needed to be tracked. The Optical Flow algorithm was used to track the movement of the bridge. Since Optical Flow is well suited for tracking small movements. This is quite important since the key points which are detected from the bridge only move a small distance. Therefore Optical Flow was ideal for the application.

3.6 Research Question 4

The application output needed to be tested. We tested the application by comparing the results with IMU sensors on a wooden bridge at the University of Twente.We compared the IMU sensor data with the application output.

3.7 Research Question 5

The stabilization of the video is done using a reference point chosen by the user. The user chooses a reference point to focus the video. The video will be calibrated when it films using this reference point. By comparing the movement per frame with the first frame. If the movement is higher than the reference point. Then the frame is translated with the difference of the xy coordinates. The frames are afterwards merged into a stabilized video.

4. RESULTS

In this section the results are shown after performing the research. Most of the programming code is written in Python and a bit of Kivy.

The first task which needed to be done was finding the best and time efficient function to detect and filter the key points. The objection detection model was trained using Yolov4.[6] After the training and testing was completed the Yolov4 weights have been created. The weights were converted to tensorflow lite.[7] The repository was used for object detection.[7] Afterwards the function for keypoints selection and keypoint tracking were combined. The application is now able to filter the key points of the bridge using the bounding box area made by TensorFlow Lite. Furthermore it is able to track the movement of the key points with the use of the Optical Flow algorithm. The movement separation is done by image translation. The video will be calibrated using the chosen point as reference. The video will be calibrated using that point. The resulting video will contain almost no mobile phone motion.

4.1 Application

The application is made using Kivy, a Python framework. The Python script can be converted to an android application. The application contains three functions. The detection of the bridge, the key points detection of the bridge and the movement estimation of the bridge key points. For the bridge detection and key point detection an image of a bridge is needed as input.

For the movement of the key points a bridge video is needed without frames of other videos or images mixed in. The user is

expected to add the files for object, keypoint detection and key point tracking inside the folders of the app.
The application works as follows:

1. Figure 17 is the homescreen. Press start to be navigated to the option screen
2. Figure 18 is the option screen. Here the user is able to choose to detect a bridge or the bridge keypoints using an image. Or detect the movement of the keypoints using a video as input. Press back to return to the option window.
3. Figure 19 is the option Image upload screen where the name needs to be ticked in and afterwards pressed to send the data for the bridge detection or keypoint detection of the bridge. Press back to return to the option window.
4. Figure 20 shows the result of the objection detection and the option to download the result. Press back to return to the option window.
5. Figure 21 shows the video input screen where the name of the video needs to be put in. Press back to return to the option window.
6. Figure 22 the user can submit the custom keypoint for stabilization and the width of the bridge separated with a comma. Press back to return to the option window.
7. Figure 23 shows that the video has been progressed and the results of a text file with the movement of the keypoints and the coordinates can be downloaded. Press back to return to the option window.
8. Figure 24 shows the result of the keypoint detection of the bridge. Press back to return to the option window.

The application was finished to be exported to an Android package. However, due to conversion issues the application was not able to be converted to an Android package.

4.2 Research question 1

The keypoint detection tool used for detecting the key points in the first frame is named:"goodFeaturesToTrack" from the opencv library. After some testing with the function on different images. The results were satisfactory. The key points selected are the best to track for the Optical Flow algorithm. Since the function has chosen the best key points to track.

4.3 Research question 2

By defining the key points the application should filter the key points. So that it only contains the key points on the bridge. This is done using the Yolov4 algorithm.[6] The training and testing were split in a ratio of 90:10 of the 500 images. The training consists of 400 images and the testing 100 images. After training the custom object detection. The following mean average precision was generated as shown in Figure 10. The mean average point(mAP) is at 86.1 percent. The Intersection over Union(IoU) threshold is set at 0.5.The result implied that the bounding box generated on the bridge is 86.1 percent accurate. It is determined by the IoU which checks per image if the bounding box generated by the model overlaps at least 50 percent with the bounding box coordinates predetermined by the user. If the overlap is 50 percent or higher than the model correctly drew the bounding box on the image. An example of the IoU threshold calculation can be found in figure 11**.**
A confusion matrix of the model can be generated by running the following command in Anaconda:
"darknet.exe detector map data/obj.data yolov4-obj.cfg backup\yolov4-obj_last.weights". The result can be observed at Figure 14. The following confusion matrix can be drawn using the results of Figure 14.

	Positive	Negative
Positive	48 (TPs)	4 (FPs)
Negative	8 (FNs)	52 (TNs)

Table 1. Confusion Matrix

The following formulas are used to calculate the recall, precision and accuracy:

Recall	$TPs/(TPs + FNs)$
Precision	$TPs/(TPs + FPs)$
Accuracy	$(TPs + TNs)/(TPs + FPs + TNs + FNs)$

Table 2. Recall, Precision and Accuracy formulas

Utilizing the formulas in Table 2 and the results in Table 1. The recall is around 0.86, the precision is around 0.92 and the accuracy is around 0.89. The accuracy differs from the maP rate due to the fact that the maP checks if the bounding box drawn on the object is within a certain threshold given by the user. Which results in an extra condition added to the TPs. This explains the different result.
Since only one bridge needed to be detected in an image the results were adequate. The weights were afterwards converted to Tensorflow Lite. Since Tensorflow Lite requires less computational power compared to Yolov4. An image is put inside the tensorflow detector to detect the bridge. If no bridge is inside the picture no bridge will be drawn. As seen in figure 9. The bounding box coordinates are saved inside a .txt file. The coordinates were used to filter out the key points. The bounding box coördinates are saved in a .txt file. The file will be used to filter the key points from research question 1. The remaining keypoints are on the bridge. The filtering can be viewed in Figure 12 and 14. Figure 12 displays the unfiltered key points. After applying the filter method with the bounding box in

Figure 13. Most of the key points outside of the bridge have been filtered out. There are some points not on the bridge due to the large area of the bounding box. However it has filtered out most of the wrong key points. So the result was sufficient.

4.4 Research question 3

The bridge movement estimation was performed by the Optical Flow algorithm.[3] The Optical Flow algorithm was used to track and calculate the movement. The Optical Flow parser the video first frame and uses the goodFeaturesToTrack to determine the keypoints. The keypoints are filtered if they remain in the bounding box using the coordinates in the .txt file. Optical Flow draws the movement per keypoint on the frame. This can be used to calculate the movement of the bridge. The code was rewritten to save the initial key points. The original key points coordinates are saved into a list. For each loop the Optical Flow algorithm will track the movement of the keypoints. The movement will be registered. With the use of p1 the new coordinates of the points. With the use of saved original key points. The distance calculation is done using the distance formula: $d = \sqrt{((x\_2 - x\_1)^2 + (y\_2 - y\_1)^2)}$.

With the formula the distance is calculated per pixel. If the distance of the next keypoint is greater than the previous distance the old distance will be replaced with the new distance. Afterwards the distance is multiplied by the scale given by the user. The scale is calculated with the use of the bounding box coordinates. The width is given by the user as input in centimeters. Afterwards the bounding box width is calculated by subtracting the right most corner minus the left most corner. The original width of the bridge is divided by the width of the bridge in pixels. Which results in the pixel scale to centimeters.

4.5 Research question 4

In order to validate the results obtained. The applications needed to be tested. The four IMU sensors were deployed on the bridge at the University of Twente. They were turned on when the person gave the signal which is seen in the video. The length of the video is around three minutes. In the video people walked over the bridge. The four IMU sensors had some connection issues. So, there is a spike in the data captured in Figure 16A,B and C. This is due to the connection loss.

As seen in Figure 16A,B and C. The spike of length was detected when the people walked over the bridge as seen in Figure 16A,B and C. IMU sensors detected will be looked at to determine the motion due to the fact that my application only returns the highest motion. The accelerometer with x,y,z data was used to estimate the displacement of the bridge. We used a displacement calculator.[1] The following inputs were used for after comparing it with the data in the .itlog file:

- X
  u = 0.001
  a = 0.002
  t = 8
- Y
  u = 0.001
  a = 0.002
  t = 8
- Z
  u = 0.001
  a = 0.006
  t = 8

The results for x and y are around 6.48 cm displacement and for z around 19.28 cm. Compared to our application a similar displacement was measured in one of the keypoints. A displacement of 5.15 in cm. With the bridge width being given as 2,5 meters. The measurement is almost on par with the IMU sensor. So, the measurement is quite okay.

4.6 Research question 5

The stabilization of the video was done using a point chosen by the user. The video is calibrated by using the chosen point as the main focus point.[9] The data gathered will be used to stabilize the video. The points are chosen on the bridge to limit the keypoint possibilities. The key point is tracked frame by frame using Optical Flow. For each frame the movement is compared to the first frame. After the video key point differences have been read. Each frame is run again. To create a stabilized video. The video is created by translating the image with the xy coordinates saved in a .txt file. The first frame is taken as the origin coordinates. Every deviation from frame 1 to the original frame will be translated accordingly. For example if the image were shifted with 50 pixels for the x and for the y 50 pixels the frame will be translated as seen in Figure 15.

The motion of the mobile is almost completely removed. Which resulted in more accurate tracking of the key points of the bridge.

5. CONCLUSION

The research had four main objectives:

- To create a key point detection tool, which can process the first frame of the video and place the key points.
- To filter the key points from goal 1. So, only the bridge key points remain. Goal 1 output will be used.
- To create a python function which will track the movement of the key points and return the movement in cm.
- To create a program to separate the movement of the camera and the movement of the bridge.

Every main objective was finished. The application is able to detect the key points in the image using the goodFeaturesToTrack(). The points are afterwards filtered using

the bounding box coordinates of the Tensorflow Lite. The movement is calculated using Optical Flow per keypoint. The separation of the movement of the phone and the bridge using the fixed point is finished. The video is stabilized by tracking a custom point and checking the movement deviation from the first frame. The other keypoints movement from the first frame will have their movement subtracted with the custom point. This results in the separation of the phone movement and the bridge movement.

The application was not completed. The object detection, keypoint detection and download button for the results have been finished. The key point tracking suffers from heavy computation time. Due to the high memory consumption, due to the Optical Flow algorithm. The application is able to progress smaller videos with a lower fps rate and duration time and there are some bugs with image loading.

6. DISCUSSION

There are quite some points which should be addressed in this research paper.

The key points detected are the best points for the Optical Flow to track. However, they are not necessarily the best key points to track for the bridge movement. Since the structural points of the bridge can be different from the key points selected by the goodFeaturesToTrack() function. This can be improved by adding an extra layer to the object detection. The interest points can be used to define them.

Second, the filter for the key points can be improved. The bounding box of the object detection removes almost all key points which are not on the bridge. However, there are some key points selected which are not on the bridge. Since the bounding box is not completely accurate since it is either a square or rectangle. Image segmentation should be able to resolve the issue. Since every pixel is color coded if it is a bridge or not. This can be done for example with image segmentation where every part of the bridge is filled in with the same color. This can be used to improve the filter.

Third, the bridge movement estimation can be improved. The implementation is quite well done. The measuring of the movement change in distance can be improved. Since the distance is measured by pixel. However, the scale of the pixel is not accurate enough.

Fourth, the application only runs on the pc. It should be able to run on Ios and Android. However, due to time constraints and conversion errors. It was not possible to arrange other options.

Fifth, the application can be improved on the efficiency of tracking the keypoints. The computation time for large videos requires more memory resources than available on the smartphone. This can be done by creating a variation of Optical Flow with low memory usage.

Sixth, the testing can be done more thoroughly for the results. Due to some issues with reading out the data and converting the estimated acceleration to speed. The testing part of the application is not on par. The test needed to be more thorough and precise.

Finally, the mobile and bridge separation can be improved. The separation focusses on a fixed point of the user. The assumption is that the fixed point will not move. However, this can happen. Which can result in slight deviation in measurements.

7. REFERENCES

[1] URL https://www.calculatorsoup.com/calculators/physics/displacement_v_a_t.php

[2] URL https://cvat.org/

[3] URL https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html

[4] URL https://docs.opencv.org/4.5.2/dc/d0d/tutorial_py_features_harris.html

[5] URL https://docs.opencv.org/4.5.2/d4/d8c/tutorial_py_shi_tomasi.html

[6] URL https://github.com/AlexeyAB/darknet

[7] URL https://github.com/haroonshakeel/tensorflow-yolov4-tflite

[8] URL mAP (mean Average Precision) for Object Detection | by Jonathan Hui | Medium

[9] URL https://streng20.wixsite.com/structuralengsite/post/measuring-bridge-vibrations-and-deflections-with-a-camera-improved

[10] URL https://unsplash.com/

[11] Amein, A., & El-Tanany, A. H. (2019). Advanced selective key point detection techniques. *Journal of Engineering Science and Military Technologies*, *3*(2), 61–69. DOI: https://doi.org/10.21608/ejmtc.2019.13509.1121

[12] Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. http://arxiv.org/abs/2004.10934

[13] Gao, Z., Lu, G., Lyu, C., & Yan, P. (2018). Key-frame selection for automatic summarization of surveillance videos: a method of multiple change-point detection. Machine Vision and Applications, 29(7), 1101–1117. DOI: https://doi.org/10.1007/s00138-018-0954-7

[14] Girdhar, R., Gkioxari, G., Torresani, L., Paluri, M., & Tran, D. (2017). Detect-and-track: Efficient pose estimation in videos. ArXiv, 350–359. DOI: https://doi.org/10.1109/cvpr.2018.00044

[15] Lu, R., & Ioannis, B. (2018). Detection of Key Components of Existing Bridge in Point Cloud Datasets. 3(June), 111–115. DOI: https://www.doi.org/10.1111/mice.12407

[16] Peters, J. F. (n.d.). Intelligent Systems Reference Library 124 Foundations of Computer Vision Computational Geometry, Visual Image Structures and Object Shape Detection. DOI: https://doi.org/10.1007/978-3-319-52483-2

[17] Shi, J., Tomasi, C. (1994). Good Features to Track. http://www.ai.mit.edu/courses/6.891/handouts/shi94good.pdf

[18] Xu, Z. (2013). Consistent image alignment for video mosaicing. *Signal, Image and Video Processing*, *7*(1), 129–135. DOI: https://doi.org/10.1007/s11760-011-0212-1

APPENDIX

A. PRELIMINARY RESEARCH

$$E(u,v) = \sum_{x,y} \underbrace{w(x,y)}_{\text{window function}} \underbrace{[I(x+u,y+v)}_{\text{shifted intensity}} - \underbrace{I(x,y)}_{\text{intensity}}]^2$$

[4]

Figure 1. E(u,v) function

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

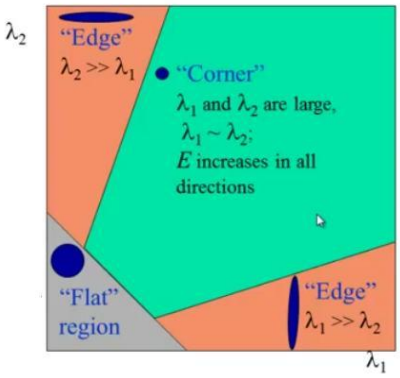
[4]

Figure 2. Taylor expansion E(u,v)

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

[4]

Figure 3. M function defined



[4]

Figure 5. Region of approval

$$R = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2$$

[4]

Figure 6. Harris Corner R function

$$R = min(\lambda_1, \lambda_2)$$

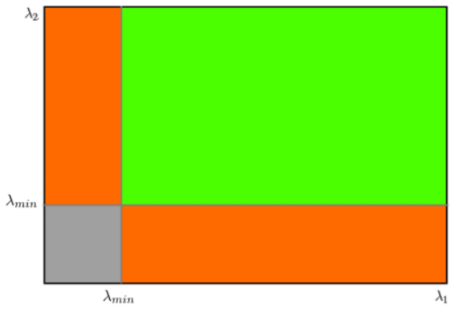
[5]

Figure 7. Shi-Tomasi redefined R function

$$R = det(M) - k(trace(M))^2$$

[4]

Figure 4. Final R function



[5]

Figure 8. Region of approval Shi-Tomasi

B. RESULTS

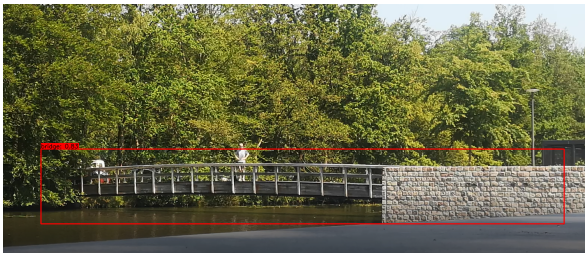


Figure 9. Map rate of Yolov4 custom bridge weights

Figure 13. Filtered Keypoints

```
mean average precision (mAP@0.50) = 0.860960, or 86.10 %
```

Figure 10. Map rate of Yolov4 custom bridge weights

```
for conf_thresh = 0.25, precision = 0.92, recall = 0.86, F1-score = 0.89
for conf_thresh = 0.25, TP = 48, FP = 4, FN = 8, average IoU = 72.73 %
```

Figure 14 Recall and precision results

[8]

Figure 11. IoU threshold example

Figure 15. Translated Image by 50 by 50 pixels

Figure 12. Keypoint detected

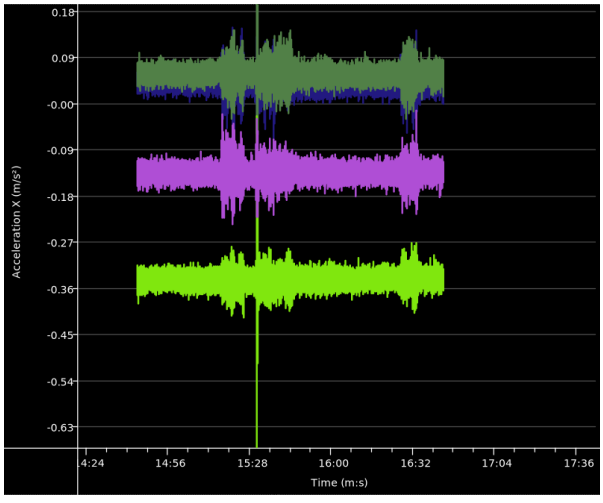


Figure 16.A A-axis accelerometer

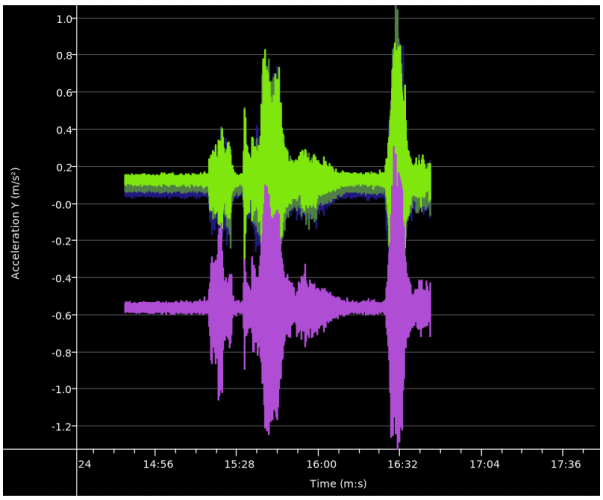


Figure 16.B Y-axis accelerometer

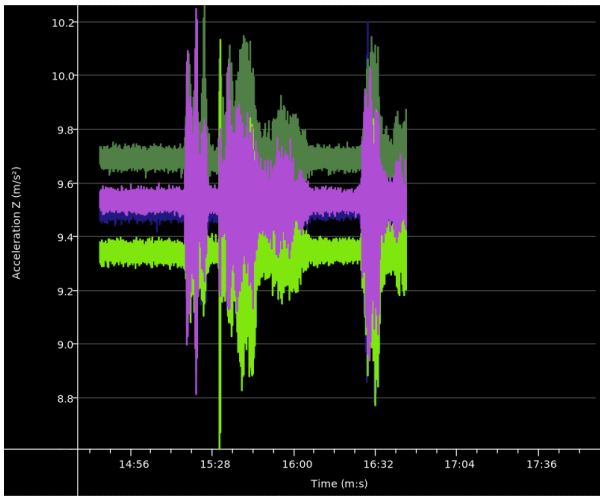


Figure 16.C Z-axis accelerometer

C. PAGES OF THE APPLICATION

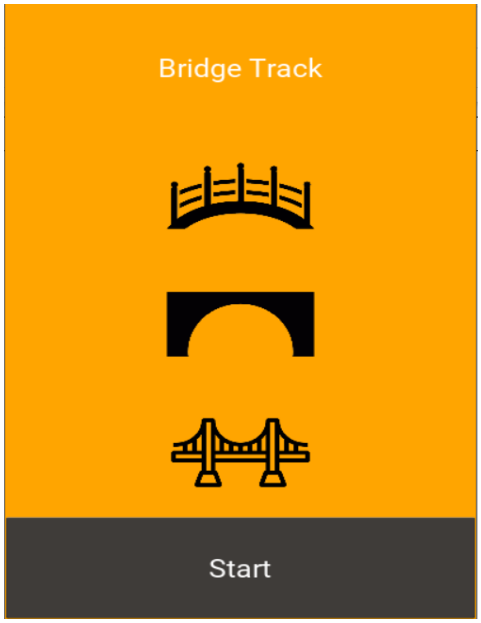


Figure 17: Home Screen

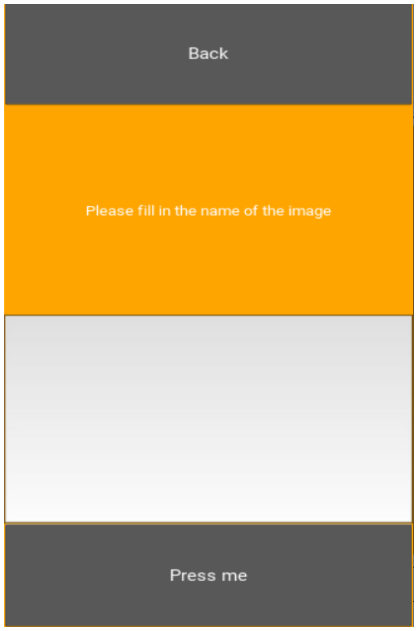


Figure 19: Image upload screen

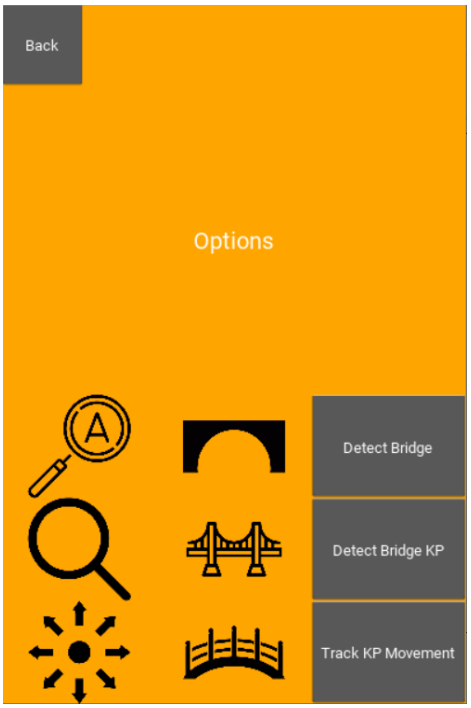


Figure 18: Option Screen

Figure 20. Result Object Detection

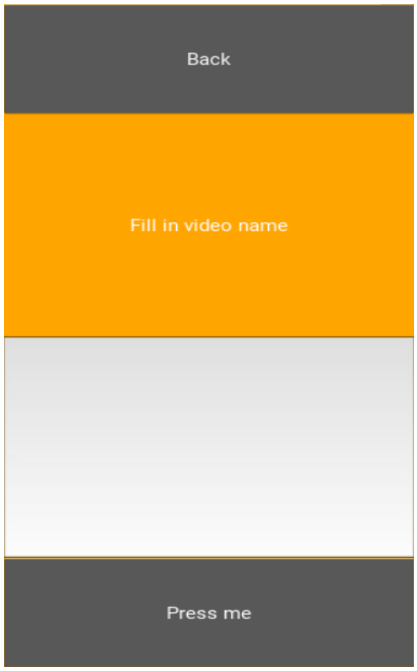


Figure 21. Video name input screen

Figure 23: Key point tracking result screen

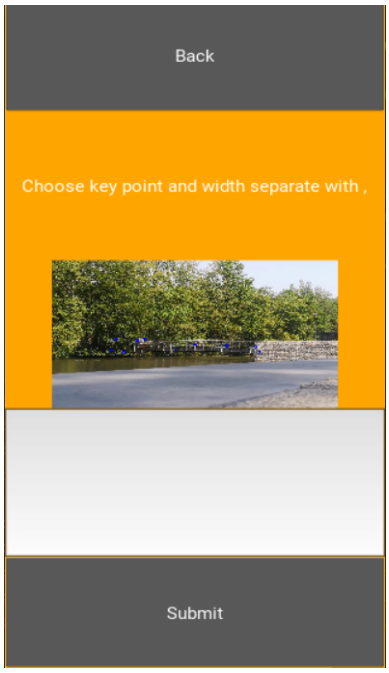


Figure 22. Choose key point screen

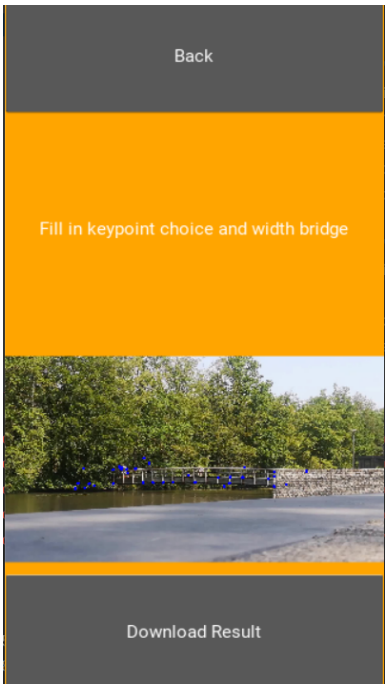


Figure 24. Result page keypoints detection bridge